Lab 8

Reg. No: 19BCE1209

Name: Gautam Sanjay Wadhwani

Course: CSE4001 Parallel and Distributed Computing

Q1. Sample Hello World

Code:

```
#include<stdio.h>

#include <mpi.h>

int main()

{

printf("hello world\n");

return 0;

}
```

Output:

```
gautam@ubuntu:~$ mpicc lab_8_1.c -o pdc
gautam@ubuntu:~$ mpirun -np 4 ./pdc
hello world
hello world
hello world
hello world
gautam@ubuntu:~$ mpirun -np 2 ./pdc
hello world
hello world
gautam@ubuntu:~$
```

Q2. Prink rank, world size and processor name

Code:

```
#include <mpi.h>

#include <stdio.h>


int main(int argc, char** argv) {

   // Initialize the MPI environment

   MPI_Init(NULL, NULL);
```

```
    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    printf("World Size: %d\n", world_size);
    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
// Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, rank %d out of %d processors\n",
        processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

Output:

```
gautam@ubuntu:~$ mpicc lab_8_2.c -o pdc
gautam@ubuntu:~$ mpirun -np 2 ./pdc
World Size: 2
Hello world from processor ubuntu, rank 1 out of 2 processors
World Size: 2
Hello world from processor ubuntu, rank 0 out of 2 processors
gautam@ubuntu:~$ mpirun -np 4 ./pdc
World Size: 4
Hello world from processor ubuntu, rank 1 out of 4 processors
World Size: 4
Hello world from processor ubuntu, rank 2 out of 4 processors
World Size: 4
Hello world from processor ubuntu, rank 0 out of 4 processors
World Size: 4
Hello world from processor ubuntu, rank 3 out of 4 processors
```

Q3. Master prints "I am Master", Worker prints "I am worker"

Code:

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char *argv[])
{
   int rank, numprocs, left, right;
   int buffer[10], buffer2[10];
   MPI_Request request, request2;
   MPI_Status status;

   MPI_Init(&argc,&argv);
   MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   if(rank == 0) printf("I am Master (rank %d)\n", rank);
   else printf("I am worker (rank %d)\n", rank);

   right = (rank + 1) % numprocs;
   left = rank - 1;
   if (left < 0)
      left = numprocs - 1;
   MPI_Irecv(buffer, 10, MPI_INT, left, 123, MPI_COMM_WORLD, &request);
   MPI_Isend(buffer2, 10, MPI_INT, right, 123, MPI_COMM_WORLD, &request2);
   MPI_Wait(&request, &status);
   MPI_Wait(&request2, &status);
   MPI_Finalize();
   return 0;
```

}

Output:

```
gautam@ubuntu:~$ mpicc lab_8_3.c -o pdc
gautam@ubuntu:~$ mpirun -np 4 ./pdc
I am worker (rank 2)
I am Master (rank 0)
I am worker (rank 3)
I am worker (rank 1)
```

Q4. Master generates 1/2,1/4,1/8,1/16...1/n; Worker generates 2,4,8,16...n

Code:

```
#include <mpi.h>

#include <stdio.h>


int rank, numprocs, left, right, n;

MPI_Request request, request2;

MPI_Status status;

void slave_method()

{

   double buffer2[n];

   buffer2[0] = 2;

   for(int i = 1; i < n; i++)

   {

         buffer2[i] = buffer2[i-1]*2;

   }

   MPI_Isend(buffer2, n, MPI_DOUBLE, right, 123, MPI_COMM_WORLD, &request2);

   MPI_Wait(&request2, &status);

   printf("Generated by slave: ");

   for(int i = 0; i < n; i++)

   {

      printf("%f ", buffer2[i]);

   }

   printf("\n");

}
```

```
int main(int argc, char *argv[])
{
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    right = (rank + 1) % numprocs;
    left = rank - 1;
    if (left < 0)
        left = numprocs - 1;
    n = 10;
    double buffer[n];
    slave_method();
    MPI_Irecv(buffer, n, MPI_DOUBLE, left, 123, MPI_COMM_WORLD, &request);
    MPI_Wait(&request, &status);
    MPI_Wait(&request2, &status);
    MPI_Finalize();
    if(rank == 0)
    {
        printf("Received from slave: ");
        for(int i = 0; i < n; i++)
        {
            printf("%f ", buffer[i]);
        }
        printf("\nGenerated by master: ");
        for(int i = 0; i < n; i++)
        {
            double value = 1.0/buffer[i];
            printf("%f ", value);
        }
        printf("\n");
```

```
    }

    return 0;

}
```

Output:

For n = 5

```
gautam@ubuntu:~$ mpicc lab_8_3.c -o pdc
gautam@ubuntu:~$ mpirun -np 4 ./pdc
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000
Received from slave: 2.000000 4.000000 8.000000 16.000000 32.000000
Generated by master: 0.500000 0.250000 0.125000 0.062500 0.031250
```

For n = 10

```
gautam@ubuntu:~$ mpicc lab_8_3.c -o pdc
gautam@ubuntu:~$ mpirun -np 4 ./pdc
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000 64.000000 128.000000 256.000000 512.000000 1024.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000 64.000000 128.000000 256.000000 512.000000 1024.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000 64.000000 128.000000 256.000000 512.000000 1024.000000
Generated by slave: 2.000000 4.000000 8.000000 16.000000 32.000000 64.000000 128.000000 256.000000 512.000000 1024.000000
Received from slave: 2.000000 4.000000 8.000000 16.000000 32.000000 64.000000 128.000000 256.000000 512.000000 1024.000000
Generated by master: 0.500000 0.250000 0.125000 0.062500 0.031250 0.015625 0.007812 0.003906 0.001953 0.000977
gautam@ubuntu:~$
```